Please amend the present application as follows:

Claims

The following is a copy of Applicant's claims that identifies language being added with underlining ("___") and language being deleted with strikethrough ("~~---~~"), as is applicable:

1. (Currently amended) A method for executing a program written for an original computer system on a different host computer system, comprising:

fetching original program instructions during execution of the program using an emulator;

interpreting the original program instructions with the emulator to determine an underlying semantic function associated with the program instructions;

providing the program instructions from the emulator to a just-in-time compiler;

growing a code fragment by linking the program instructions to the code fragment using the just-in-time compiler;

dynamically translating the original program instructions of the code fragment with the just-in-time compiler to generate ~~into~~ instructions that can be executed by the host computer system;

dynamically emitting translated program instructions from the just-in-time compiler into at least one code cache of a dynamic execution layer interface of a virtual machine; and

dynamically executing the translated instructions from the at least one code cache in lieu of the original program instructions when a semantic function of the original program instructions is requested.

2-4. (Canceled)

5. (Previously presented) The method of claim 1, wherein emitting translated program instructions into at least one code cache comprises emitting translated program instructions into the at least one code cache via an application programming interface.

6. (Previously presented) The method of claim 1, further comprising interpreting and executing program instructions that have not be emitted into the at least one code cache.

7. (Previously presented) The method of claim 6, further comprising emulating exception handling actions that would have been performed by the original computer system during execution.

8-19. (Canceled)

20. (Previously presented) A system for executing program code that was written for an original computer system on a different host computer system, comprising:

an emulator configured to fetch original program instructions during execution of a program and dynamically interpret the original program instructions to determine an underlying semantic function associated with the program instructions;

a just-in-time compiler configured to receive the program instructions from the emulator, dynamically grow a code fragment by adding the program instructions to the code fragment, and dynamically translate the original program instructions into instructions that can be executed by the host computer system;

a virtual machine that comprises a dynamic execution layer interface including a core having at least one code cache in which code fragments can be dynamically cached and executed; and

an application programming interface that links the ~~translator~~ just-in-time compiler to the virtual machine, the application programming interface being configured to emit instructions that have been translated by the just-in-time compiler to the code cache for execution of the translated instructions in lieu of the original program instructions.

21-22. (Canceled)

23. (Original) The system of claim 20, wherein the application programming interface comprises a set of functions available to the translator including an emit fragment function with which the translator can emit code fragments into the at least one code cache and an execute function with which the translator can request execution of code fragments contained within the at least one code cache.

24. (Original) The system of claim 23, wherein the application programming interface is configured such that the emit fragment function can be used to perform at least one of tracking a cached code fragment and associating metadata with a cached code fragment.

25. (Original) The system of claim 23, wherein the set of application programming interface functions comprises a lookup fragment function with which cached code fragments can be retrieved.

26. (Original) The system of claim 23, wherein the set of application programming interface functions comprises an invalidate fragment function with which individual cached code fragments can be invalidated.

27. (Original) The system of claim 23, wherein the set of application programming interface functions comprises a enumerate fragment function with which cached code fragments can be enumerated using metadata associated with the fragments.

28. (Original) The system of claim 23, wherein the set of application programming interface functions comprises a cache flush function in which code caches of the dynamic execution layer interface can be flushed.

29. (Original) The system of claim 23, wherein the set of application programming interface functions comprises an install callback function with which the translator can be notified as to events that occur within the dynamic execution layer interface.

30-36. (Canceled)

37. (Previously presented) The method of claim 1, wherein fetching original program instructions comprises accessing original memory addresses to identify actual locations of the instructions on the host computer system.

38. (Previously presented) The method of claim 1, wherein dynamically executing translated instructions comprises dynamically executing translated instructions within the at least one code cache until a cache miss occurs.

39. (Previously presented) The method of claim 38, further comprising fetching other original program instructions, dynamically translating the other original program instructions, dynamically emitting further translated program instructions into the at least one code cache, and dynamically executing the further translated program instructions within the at least one code cache.

40. (Previously presented) The method of claim 39, further comprising repeating the procedures of claim 39 until substantially all execution of the program comprises execution of translated program instructions within the at least one code cache.